DO DESENVOLVIMENTO PARA A IMPLANTAÇÃO

ENGENHARIA DE SISTEMAS DE INFORMAÇÃO

Daniel Cordeiro

1º de dezembro de 2017

Escola de Artes, Ciências e Humanidades | EACH | USP

AGENDA

- Disponibilidade & Responsividade
- Apdex
- Monitoramento
- Atualizações & flags de funcionalidades
- Atenuando a pressão sobre o banco de dados: índices & consultas abusivas
- Atenuando a pressão sobre o banco de dados: uso de cache
- Protegendo os dados do cliente

DESENVOLVIMENTO VS. IMPLANTAÇÃO

Desenvolvimento
Testes para garantir que seu app funciona tal como projetado.

Implantação
Testes para garantir que seu app funciona quando usado de forma para o qual ele não foi projetado para ser usado.

MÁS NOTÍCIAS

- “Usuários são seres terríveis”
- Alguns bugs só aparecem quando o sistema está sob estresse
- Ambiente de produção ≠ ambiente de desenvolvimento
- O mundo está cheio de forças do mal
- … e de idiotas

BOAS NOTÍCIAS: PAAS DEIXOU A IMPLANTAÇÃO MUITO MAIS FÁCIL

- arrume um Servidor Virtual Privado (*Virtual Private Server* ou VPS), talvez em uma plataforma de computação em nuvem
- instale & configure Linux, Rails, Apache, *mysqld*, openssl, sshd, ipchains, squid, qmail, logrotate, ...
- corrija (quase que toda semana) as vulnerabilidades de segurança
- descubra que você se encontra em uma *Library Hell*
- ajuste tudo que puder para conseguir o máximo possível por cada $ investido
- descubra um jeito de automatizar a escalabilidade horizontal

NOSSO OBJETIVO: SE MANTER A UM PAAS

PaaS gerencia...	Nós gerenciamos
As camadas “fáceis” de conseguir escalabilidade horizontal	Minimização da carga no banco de dados
Ajustes no desempenho dos componentes do sistema	Ajustes no desempenho da aplicação (ex: caching)
Segurança no nível da infraestrutura	Segurança no nível da aplicação

Mas isso é factível na prática?

- Pivotal Tracker & Basecamp rodam cada um em um único BD (computador commodity de 128GB < US$ 10 mil)
- Muitos apps SaaS não operam em escala global (interno ou de interesse limitado)
- DevOps is dead. Long live DevOps![1]

[1] https://techcrunch.com/2016/04/07/devops-is-dead-long-live-devops/

DESEMPENHO & SEGURANÇA

- Disponibilidade ou *uptime*

  Qual % do tempo em que o site está no ar & acessível?
- Responsividade

  Quanto tempo demora do clique do usuário até ele ver a resposta?
- Escalabilidade

  A medida que o # usuários aumenta, você consegue manter a responsividade sem aumentar o custo/usuário?

- Privacidade

  O acesso aos dados é limitado apenas aos usuários apropriados?
- Autenticação

  Podemos confiar que o usuário é quem ele diz ser?
- Integridade de dados

  É possível perceber uma violação dos dados mais sensíveis do usuário?

PERGUNTA

Sejam:

- **E** a disponibilidade do EP de vocês
- **H** a disponibilidade do Heroku
- **C** a disponibilidade de conexão a Internet
- **P** a percepção do professor sobre a disponibilidade do EP

Qual afirmação é verdadeira?

1. $P \leq C \leq H \leq E$
2. $P \geq \mathbf{min}(C, H, E)$
3. $P \leq C \leq \mathbf{min}(H, E)$
4. Não dá para saber sem informações adicionais

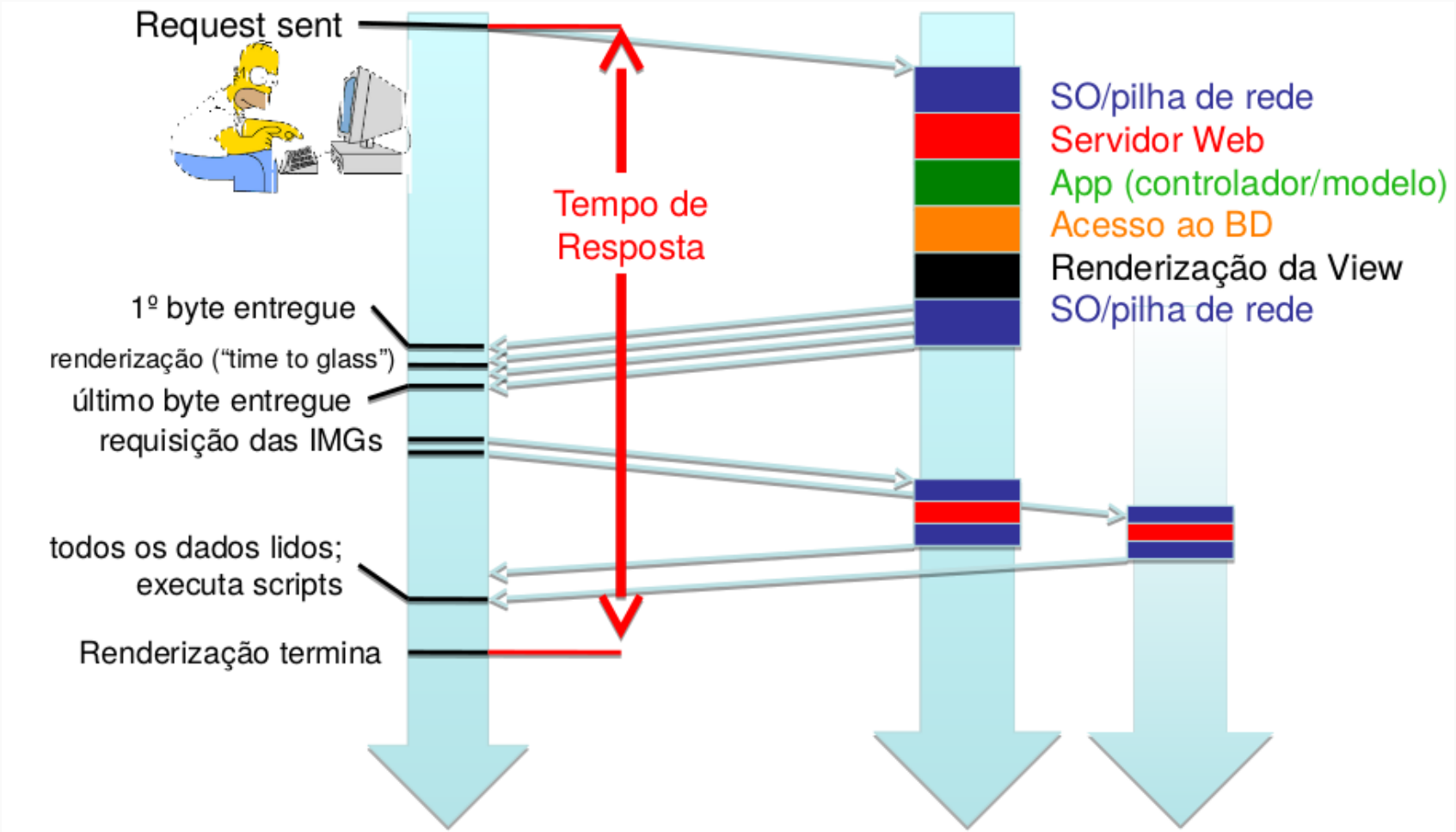
Quantificando a Disponibilidade & Responsividade

O tempo de resposta é importante?

- Quão importante é o tempo de resposta?[2]
  - Amazon: +100ms → queda de 1% nas vendas
  - Yahoo!: +400ms → queda de 5–9% do tráfego
  - Google: +500ms → 20% menos buscas
- Estudos clássicos (Miller, 1968; Bhatti, 2000)
  - $<$ 100ms é "instantâneo"
  - $>$ 7s já é hora de desistir
- `http://code.google.com/speed`

[2] Fonte: Nicole Sullivan (Yahoo! Inc.), Design Fast Websites, `http://www.slideshare.net/stubbornella/designing-fast-websites-presentation`

PARA ONDE O TEMPO VAI (SERVIDOR/REDE)?



PARA ONDE O TEMPO VAI (CLIENTE)?

- Seletores CSS + Interpretador JavaScript = 41% do tempo de renderização no cliente
  - especialmente seletores que requerem percorrer a árvore DOM (ex: `div > li`)
- Navegadores competem na velocidade de seus interpretadores JavaScript ⇒ desempenho do seletor/parser é frequentemente o gargalo

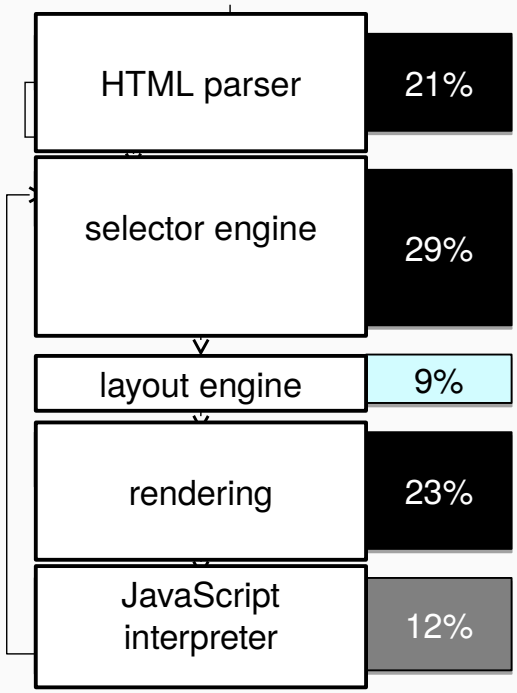
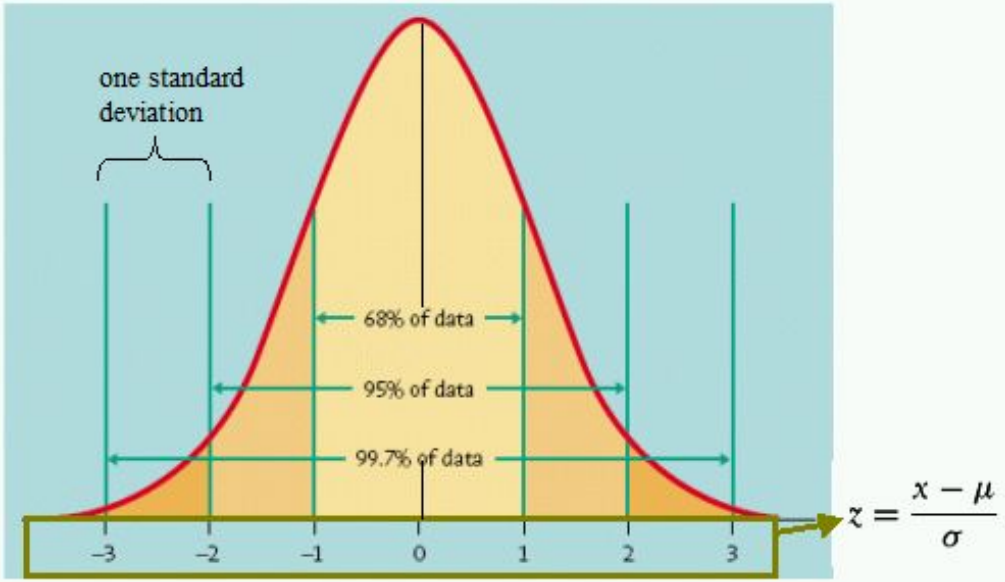


Figura 1: Cortesia de Leo Meyerovich, UC Berkeley

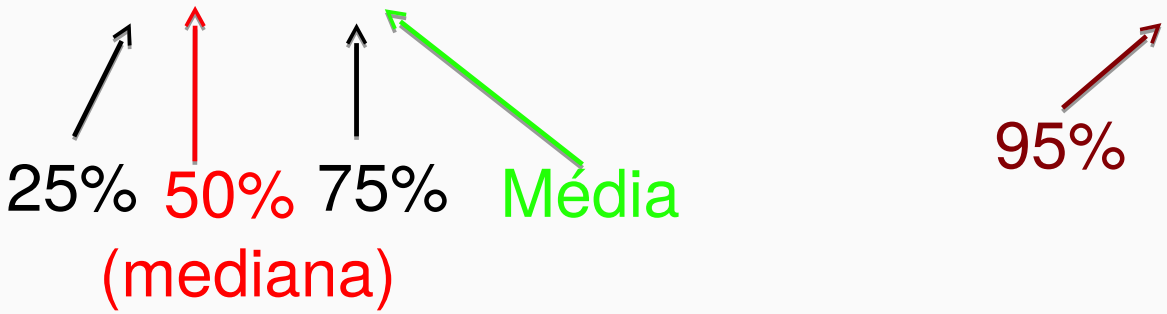
VISÃO SIMPLIFICADA (E FALSA) DO TEMPO DE RESPOSTA

- Com tempos de resposta seguindo uma *distribuição normal* em torno da média: o tempo de resposta está em média ± 2 desvio padrão com 95% de confiança
- Tempo de resposta *médio* T significa que:

- 95% dos usuários recebem $T + 2\sigma$
- 99,7% dos usuários recebem $T + 3\sigma$



UM EXEMPLO REAL



Courtesia de Bill Kayser, Distinguished Engineer, New Relic.
`http://blog.newrelic.com/breaking-down-apdex`.
Usado com permissão do autor.

PERGUNTA

Quais das opções seguintes é a *menos provável*, na maior parte dos casos, para melhorar o tempo de renderização/tempo de exibição para um app SaaS como o do EP de vocês?

1. Usar seletores simples, tal como `#id`, ao invés de usar seletores aninhados, tal como `p > span`
2. Usar técnicas de estilo CSS ao invés de usar JavaScript para implementar coisas como “navegação por abas” ou animações quando o ponteiro do mouse estiver por cima de um elemento
3. Aumentar a largura de banda efetiva entre o servidor e o cliente
4. Usar o *Rails asset pipeline* para minificar o JavaScript e concatenar todo código JavaScript em um único arquivo